Lagrangian Approximations for the Stochastic Reachability of a Target Tube

Table of Contents

This example will demonstrate how to use the SReachTools toolbox to compute over and under approximations for the stochastic reachability of a target tube via Lagrangian methods.

Lagrangian Methods

Lagrangian methods perform computations with sets using operations like unions, intersection, Minkowski addition/differences, etc. This computation using set operations can be used to approximate (either over or under) the stochastic reachability of a target tube problem. We will demonstrate that this approach while being be approximative can outperform the current state-of-the-art dynamic programming solution in terms of computation time.

Advantages:

- No gridding, which partially evades the curse of dimensionality
- Provides verification for closed-loop feedback strategies

Disadvantages:

- Using Polyhedral representation, must solve the vertex-facet enumeration problem, limiting computations to ~4 dimensional systems
- Does not provide an explicit control policy, only verifies the existence

The theory for this approach can be found in

- J. D. Gleason, A. P. Vinod, M. M. K. Oishi, "Underapproximation of Reach-Avoid Sets for Discrete-Time Stochastic Systems via Lagrangian Methods," in Proceedings of the IEEE Conference on Decision and Control, 2017.
- J. D. Gleason, A. P. Vinod, M. M. K. Oishi, "Underapproximation of Reach-Avoid Sets for Discrete-Time Stochastic Systems via Lagrangian Methods," in Proceedings of the IEEE Conference on Decision and Control, 2017

Problem Definition

In this example we will look at the viability problem for a double integrator. The system dynamics are:

$$x_{k+1} = \left[\begin{array}{cc} 1 & T \\ 0 & 1 \end{array}\right] x_k + \left[\begin{array}{c} \frac{T^2}{2} \\ T \end{array}\right] u_k + w_k$$

where $x_{k+1} \in \mathbf{R}^2$ is the state, $u_k \in \mathbf{R}$ is the input, and $w_k \in \mathbf{R}^2$ is the disturbance. The following code defines this system with w_k as an i.i.d. Gaussian disturbance with mean $[0, 0]^\top$ and variance $\mathrm{diag}(0.01, 0.01)$.

example parameters

```
T = 0.25;

% define the system
sys = getChainOfIntegLtiSystem(2, ...
    T, ...
    Polyhedron('lb', -0.1, 'ub', 0.1), ...
     RandomVector('Gaussian', zeros(2,1), 0.001*eye(2)));
```

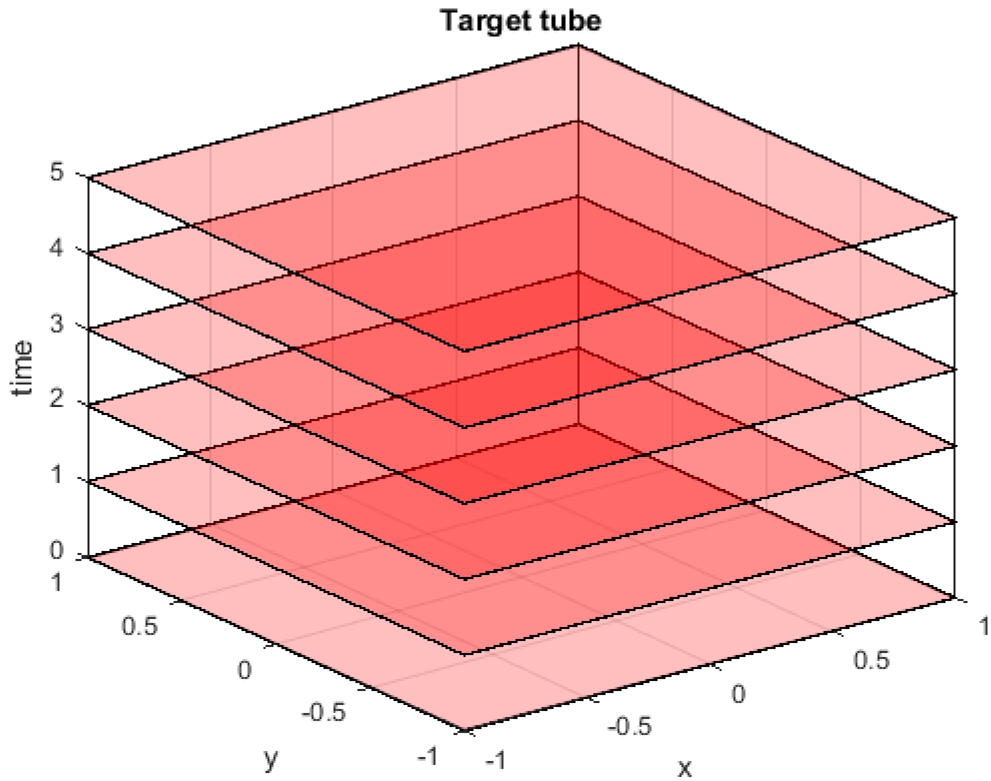
Viability problem as a stochastic reachability of a target tube problem

We examine the viability problem in which we are interested in staying in a set of safe states. In this example the safe set is $\{x \in \mathbf{R}^2 : |x_i| < 1, i = 1, 2\}$. The stochastic reachability of a target tube problem posed as a viability problem by constructing a target tube in which all sets in the tube are the safe set.

```
time_horizon = 5;

% safe set definition
safe_set = Polyhedron('lb', [-1, -1], 'ub', [1, 1]);
% target tube definition
target_tube = Tube('viability', safe_set, time_horizon);
% probability threshold desired
beta = 0.8;
% Plotting of target tube
figure()
hold on
for time_indx = 0:time_horizon
    target_tube_at_time_indx = Polyhedron('H',[target_tube(time_indx
+1).A,zeros(size(target_tube(time_indx+1).A,1),1),
 target_tube(time_indx+1).b], 'He',[0 0 1 time_indx]);
    plot(target_tube_at_time_indx, 'alpha',0.25);
end
axis([-1 1 -1 1 0 time_horizon]);
box on;
grid on;
```

```
xlabel('x');
ylabel('y');
zlabel('time');
title('Target tube');
```



Lagrangian approximation for stochastic reachability of a target tube

For the Lagrangian methods we compute robust and augmented effective target sets---for the under and overapproximations, respectively. For this computation we need to convert the Gaussian disturbance into a bounded distrubance set which will satisfy the required conditions detailed in the aforementioned papers. We do this here for a an 0.8 probability with the given target tube.

There are several ways to create bounded disturbance sets. Here, we formulate a bounded disturbance by creating a polyhedral approximation of an ellipsoid through random direction choices.

bounded set for Lagrangian

```
tic;
opts = SReachSetOptions('term', 'lag-under', 'bound_set_method', ...
    'ellipsoid');
luSet = SReachSet('term', 'lag-under', sys, 0.8, target_tube, opts);
lagrange_under_time = toc();

tic;
```

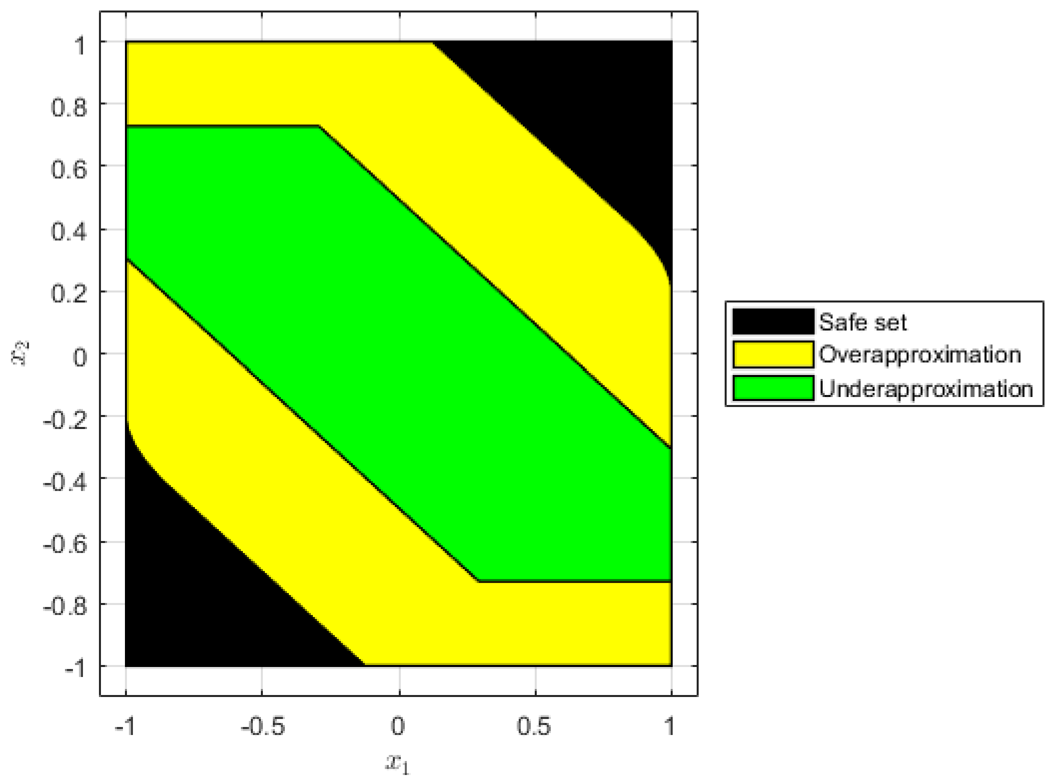
```
opts = SReachSetOptions('term', 'lag-
over', 'bound_set_method', 'random', ...
    'num_dirs', 50);
loSet = SReachSet('term', 'lag-over', sys, 0.8, target_tube, opts);
lagrange_over_time = toc();
```

Now we can compute the Lagrangian under and overapproximations which we call the robust and augmented effective target sets

```
% robust_eff_target = getRobustEffTarget(sys, target_tube,
 lag_bounded_set);
% aug_eff_target    = getAugEffTarget(sys, target_tube,
 lag_bounded_set);
```

Plotting these sets

```
figure();
plot(safe_set, 'color', 'k', 'alpha',1);
hold on;
plot(loSet, 'color', 'y');
plot(luSet, 'color', 'g');
hold off;
xlabel('$x_1$', 'Interpreter', 'latex')
ylabel('$x_2$', 'Interpreter', 'latex')
box on;
leg = legend('Safe set','Overapproximation','Underapproximation');
set(leg,'Location','EastOutside');
```



Because of the choice or random directions for the ellipse `robust_eff_target` and `robust_target_2` are not exactly equivalent (same for the augmented sets). However they can be seen to be visually near identical.

Dynamic programming solution

We can compare the results with dynamic programming to see how the approximations appear and how they compare in simulation times.

```
dyn_prog_xinc = 0.025;
dyn_prog_uinc = 0.1;
tic;
[prob_x, cell_of_xvec] = SReachDynProg('term', sys, dyn_prog_xinc, ...
    dyn_prog_uinc, target_tube);
dynprog_time = toc();
```

Compute the 0.8- stochastic level set

```
dyn_soln_lvl_set = getDynProgLevelSets2D(cell_of_xvec, prob_x, 0.8,
 target_tube);
```

Simulation times --- Lagrangian approximation is much faster than dynamic programming

The simulation times for Lagrangian computation is much faster than dynamic programming, even when the former computes both an underapproximation and an overapproximation.

```
fprintf('Simulation times [seconds]:\n');
fprintf('    Lagrangian:\n');
fprintf('        Overapproximation  : %.3f\n', lagrange_over_time);
fprintf('        Underapproximation : %.3f\n', lagrange_under_time);
fprintf('    Dynamic programming    : %.3f\n', dynprog_time);
```

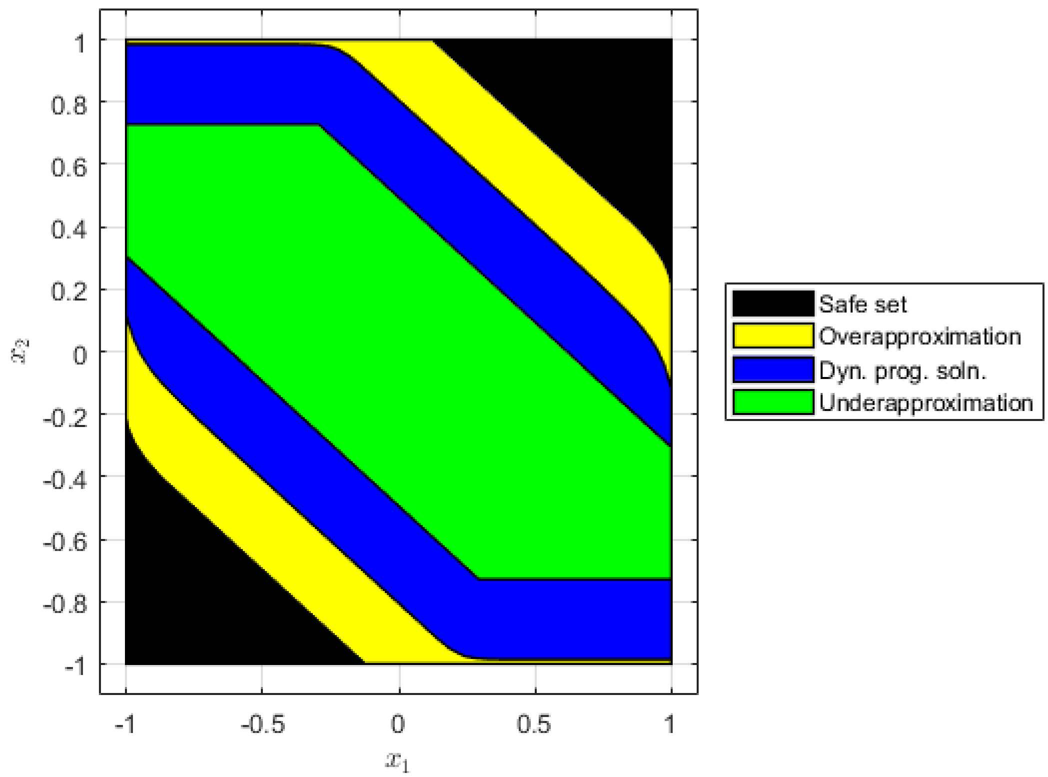
```
Simulation times [seconds]:
    Lagrangian:
        Overapproximation  : 0.567
        Underapproximation : 0.427
    Dynamic programming    : 13.324
```

Plotting all the sets together

As expected, the over-approximation and the under-approximation obtained via Lagrangian approach bounds the dynamic programming solution from "inside" and "outside".

```
figure();
plot(safe_set, 'color', 'k', 'alpha',1);
hold on;
plot(loSet, 'color', 'y');
plot(dyn_soln_lvl_set,'color', 'b')
plot(luSet, 'color', 'g');
hold off;
xlabel('$x_1$', 'Interpreter', 'latex')
```

```
ylabel('$x_2$', 'Interpreter', 'latex')
leg = legend('Safe set','Overapproximation', 'Dyn. prog.
 soln.','Underapproximation');
set(leg,'Location','EastOutside');
box on;
```



Published with MATLAB® R2017a